Association for Information Systems
AIS Electronic Library (AISeL)

ACIS 2006 Proceedings | Australasian (ACIS)

2006

A Web Services Architecture for Rich Content Mobile Learning Clients

David Parsons
Massey University, d.p.parsons@massey.ac.nz

Joshua Newnham
Massey University

A Web Services Architecture for Rich Content Mobile Learning Clients

Dr. David Parsons
Joshua Newnham
Massey University

Institute of Information and Mathematical Sciences
Massey University
Auckland, New Zealand
Email: d.p.parsons@massey.ac.nz

Abstract

The increasing use of mobile platforms for application deployment provides new opportunities for information systems architects, but with these opportunities come a number of challenges. The wide range of different types of mobile device, their operating systems, form factors, language runtimes and browser markup make it difficult to deploy generic mobile applications that work reliably and efficiently in all situations. It seems that in many cases there must a trade off between providing a quality user experience and delivering content that is not tailored to a specific type of device. However there are some software tools and architectures that can help us to minimise this trade off. In this paper we evaluate an approach to this problem using a Java Micro Edition based smart client, given generic presentation properties via J2ME Polish and communicating in a platform neutral way using XML messaging. The context of the prototype implementation is a mobile learning system that uses a range of media types in its presentation. This paper describes the software architecture, the key design and implementation issues and an analysis of the performance of the system in realistic connection contexts.

Keywords

Mobile web services, Java Micro Edition, XML, mobile learning

INTRODUCTION

Mobile web based applications may be engineered using a variety of approaches, each of which has its own strengths and weaknesses. These approaches include thin client mark-up, rich browser client and smart client applications. As the power and flexibility of mobile devices increases, there are more architectural options for developing mobile applications, and user's expectations also increase.

In this paper we describe a mobile web-based application that uses a lightweight XML (eXtensible Markup Language) over HTTP (Hypertext Transfer Protocol) communication protocol and a Java Micro Edition (Java ME) smart client[*]. The motivation for this design was to enable the flexible inclusion of rich content into a programmable but platform independent client, in order to deliver a relatively rich client experience to a wide range of devices. Although Java ME has its limitations compared to some other smart client platforms such as Symbian and BREW, it works across a large proportion of mobile devices (Coulton et al. 2005), including many different Personal Digital Assistants (PDAs), Blackberry devices and mobile phones. Three quarters of the mobile phones shipped in 2005 included Java support (Mahmoud 2005). One major advantage that Java ME has over some other tools is that it is hosted by phones using a range of operating systems, including Windows Mobile. This means that such applications will be able to continue to be deployed widely even if the operating system market changes significantly, as some have suggested it might (WindowsForDevices.com 2006). In the context of mobile phones, the specific configuration and profile typically installed is the Connected Limited Device Configuration (CLDC) supporting the Mobile Information Device Profile (MIDP). Java ME applications that run using this profile are known as MIDlets. Of course coding at the higher level of abstraction that enables interoperability has its costs in performance terms. For example Java ME applications have been shown to execute at about half the speed of equivalent C programs (Domer et al. 2004). In addition, transmitting and processing information in XML format is also relatively inefficient. Therefore one of the issues addressed in this paper is how such a system performs in a realistic deployment context.

Mobile learning

Mobile Learning is gaining popularity as a way of providing learners with educational material wherever they are and at any time. With new capabilities continually being added to mobile devices, one major challenge, and opportunity, is finding innovative ways to enhance the learning experience using these new technologies. Mobile

[*] Java ME was formerly known as Java 2 Micro Edition (J2ME)

learning is not just e-learning on a mobile device, since much e-learning content is inappropriate for small devices (i.e. small screens make it hard to read a lot of text) as well as inappropriate for the typical context that these devices will be used in. In many mobile learning scenarios the device will be used on casual basis or as a tool to obtain knowledge when required (Koschembahr and Sagrott 2005). Therefore course material must be packaged in short and focussed learning units. In this paper we explore one possible approach to realizing a mobile learning framework, using Java ME and XML to deliver rich and interactive content to the mobile learner. We describe the architecture of the prototype system developed; the overall framework, the communication mechanism used between the client and server, the data model used to capture learning content and student progress information, and aspects of the user interface. We also provide some test results from various deployment configurations to assess how well this architecture performs in practice in terms of user waiting time.

Related work

Although there has been a large amount of work published on mobile learning, little has been focused on the type of architecture we describe here. (Sakkopoulos et al. 2006) describe a mobile learning system using a form of web services based on SMS. The authors address issues of browser adaptivity but in this example XML is not used as the transport mechanism. (Chang 2006) considers a range of server side technologies for delivering learning technology, and notes the benefits of service oriented architecture, though does not provide any explicit architecture but reviews user testing of third party implementations. MX-Learn appears to have some similarities in terms of XML data transmission, and includes some compression facilities, but the architecture of the client is not explicitly provided, and it appear that the primary XML processing takes place on the server rather than the client (Casalino et al. 2006). Another platform that has similarities with the one described here is the Mobile Learning Engine, which also uses a Java ME smart client and XML, but has different deployment versions for different types of mobile phone (Meisenberger 2004). Since available documentation in English is limited, it has not been possible to fully analyse the architecture of this system.

TECHNICAL ARCHITECTURE

In this section we outline the rich client mobile learning framework. First we discuss the overall goals of such a framework, and then examine the process of a client request followed by a discussion about communication protocol trade-offs, and finally discuss some user interface issues.

Framework Overview

The basic requirements for the system were to provide a flexible framework that could easily be extended (by adding new content types), adapted (to different platforms) and provide a rich and interactive environment that could potentially enhance learning.

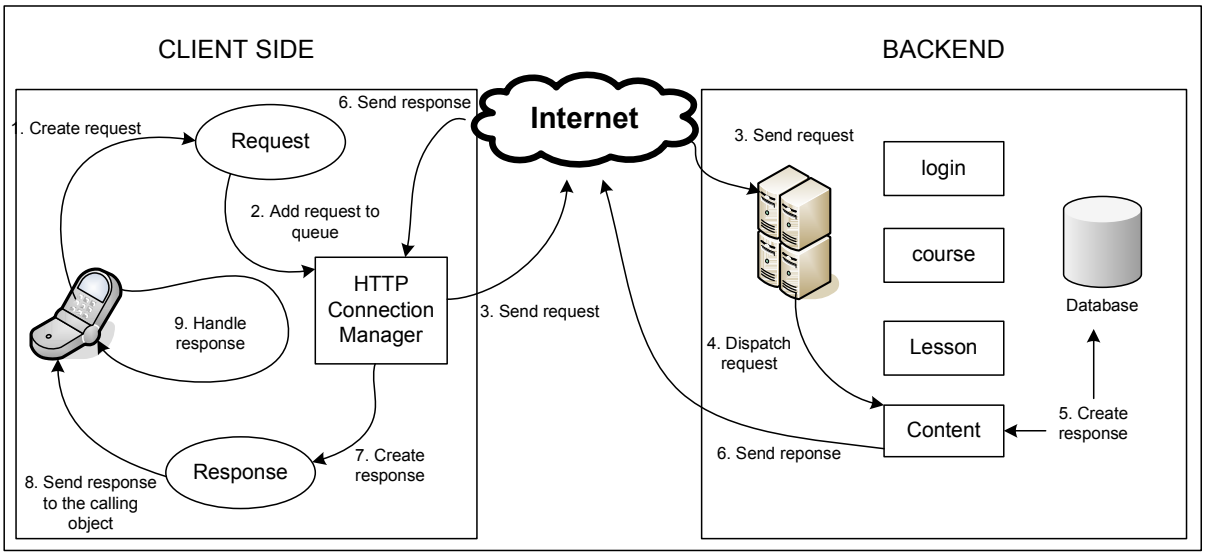

Figure 1: Client server interaction process using XML over HTTP

We begin by describing the approach used to process client requests. Figure 1 shows the process of the client interacting with the server using XML over HTTP. Server side components are primarily Java servlets. The process begins with the Java ME client creating a request object containing information such as destination servlet, servlet operation and associated parameters. Once the request object has been populated, the client sends

this request off to the *HTTP connection manager*, a component that acts as a client-side proxy for the server by assembling and disassembling the XML messages. From here the *HTTP connection manager* passes the request off to the server as an XML message and listens for a response. The XML message is received by the *Dispatcher* servlet that takes the message and de-serializes it into a request object; once this is done it then passes the request object to the appropriate servlet for dealing with the user's content request. This servlet handles the request, generates a response object by accessing the data store via a layer of data access objects, and finally sends this response object back to the client. The response is received by the client's *HTTP connection manager* which then de-serializes the XML response into an appropriate response object and passes this back to the calling object.

Server Side components

The components on the server side are primarily Java servlets, acting as controllers and command objects to process client requests. Apart from the MIDlet download page, server pages are not used in the application since page presentation is not required. The four major components on the server include a dispatcher servlet that acts as a front controller (Fowler 2003), a number of servlets specific to request types, which act as the command or action components of the front controller pattern, a data access layer that consists of factory and provider object that utilise the Hibernate object relational mapping framework, and finally the actual data objects that are mapped to the database. The dispatcher servlet's responsibility is to handle the initial request, de-serialize the XML data into a request object and pass the request off to the appropriate servlet. Figure 2 shows these components mapped into a front controller pattern, with the special characteristic that the key role of these servlets is to process XML requests and responses.

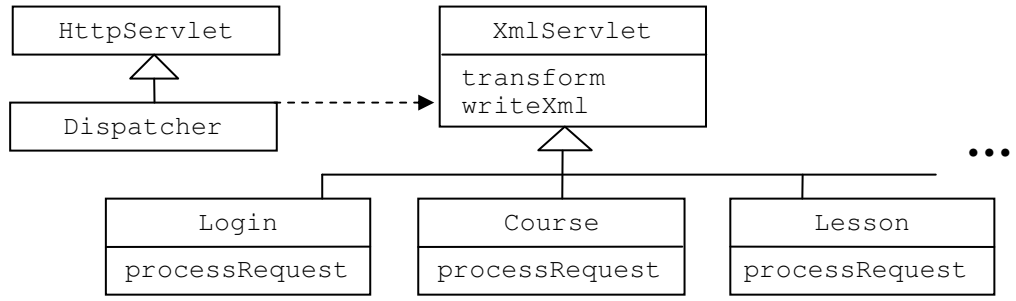

Figure 2: Implementation of the front controller pattern that processes XML requests and responses

Each request type specific servlet handles a different type of request from the user; essentially the business logic is packaged into these command classes. The request object will contain the requested method name and associated parameters. The servlet queries this request object and executes the appropriate method. Most of these methods rely on requesting information so the servlet will communicate with the data access layer to obtain the required information. Each data object implements the XMLGenerator interface that contains the method signature **getXML()**. This method is what the servlet uses to generate the XML from the data objects. Once the servlet has the appropriate XML document it will pass it back to the client.

Data Model

Storing content in the database is a standard design pattern for dynamic web applications that have content that may frequently change (Graham 2003). The system developed here uses the MySQL relational database (MySQL AB 2006). Mapping between the server-side object model and the database is implemented using the Hibernate Core object relational mapping tool (Hibernate 2006). For the mobile learning application the three major entities in the data model are **Student**, **Course**, and **Course Content**. The student entity includes login information as well as user progress data, while course information includes available courses, course lessons, and available content. The part of the model that encapsulates course content provides support for extending the set of content types that can be used in the application. The current content types that are supported include **text** (basic informative text for a particular lesson), **rich media** such as video or audio and **multi-choice** content that allows for testing the user. Some work is required for both enhancing the existing content types (e.g. including images in the text type and including images and media for multi-choice questions to allow the teacher to present questions based on rich content) as well as creating new content types, such as other interactive activities. Having an extensible model means that we can add new types of content, and using XML as the transport mechanism means that external resources such as RSS (Really Simple Syndication) feeds and web services can be consumed by the server and delivered to the client directly. Therefore the use of XML as the transport mechanism opens up the possibility of integrating mashup web services into the application in a Web 2.0 style.

Communication with XML over HTTP

The main reason for selecting XML as the message format was because of its flexibility, enabling different types of client to reuse the same content from the server (Hjelm 2000). Not only is XML used to provide services to a smart client, but the same document could be transformed using XSLT on the server to provide content for thin client bowsers. Using HTTP as the transfer protocol guarantees that a large number of devices will be able to act as clients to the server, even mobile phones using MIDP version 1.0. Although some Java ME devices may be able to communicate using other protocols such as the User Datagram Protocol (UDP), they are less likely to be supported than HTTP since only HTTP is a required MIDP connection type within the Generic Connection Framework (Ortiz 2003).

The XML processing on the client is done using the kXML parser, an XML pull parser that avoids the fragility of stream based SAX parsers while being more memory efficient than tree-based DOM parsers (Slomiski 2004). It should be noted that this system does not use web service protocols such as the Simple Object Access protocol (SOAP), since these would incur a signicant processing and transport overhead. Since interoperability with external systems was not a major requirement for this system, the benefits of using a more lightweight XML format were felt to outweigh the benefits of using standardised XML protocols. There are still, however, two potential performance problems evident in using XML as the messaging protocol. First, it increases the processing required on the client (for serializing and de-serializing objects for transportation) compared to, for example, using a thin client Wireless Markup Language (WML) or XHTML-Mobile Profile browser. Examining the Network Analyzer provided with the J2ME Wireless Toolkit showed that the processing of the data was more time consuming than the transportation of the data between the client and server when the client and server were running on the same node. The second problem is that using XML increases the data being transported across the network, because XML's self-describing properties render it relatively verbose. This is a problem for two reasons, the first being that there may be an increase in cost (in monetary terms) for users who may be charged on the basis of bytes transferred. The second problem this method imposes is that there is an increase in upload traffic on the network, since the client sends an XML document to the server rather than a standard HTTP request. Some solutions for these issues include using a different communication protocol (i.e. using standard HTTP GET requests with parameters) except when an external web service is being exposed or used, or possibly using binary compression on the XML before transmission. This method is used by the Wireless Markup Language (WML) to reduce the size of documents transmitted over the Wireless Access protocol (WAP) by using a compact binary representation, the WAP Binary XML Content Format (WBXML) (WAP Forum 1999). In this case you could still have the flexibility of XML by adding an additional layer to the server that acts as a proxy between mobile client requests and the server. In the current prototype XML documents are transmitted as uncompressed text, but the request mechanism in the framework has been abstracted to allow for easy adoption of different communication protocols.

Given these issues, one focus of this research was to examine the effectiveness and efficiency of the chosen communication protocol. The factors that we considered were:

- The size of the client side application (MIDlets have to be installed on small devices with limited memory that is shared by other applications, pictures, games etc.)
- The self descriptiveness of the data, enabling client side processing
- The bandwidth consumption of the content types being transported and the resulting speed of the upload and download
- The processing power required and how that might impact on different client platforms

Figure 3 compares the different types of communication protocols available and the trade-offs between them. In essence, our XML usage can be seen as a compromise between a number of forces.

One of the further considerations that can influence our choice is how well a given technology integrates with particular technical platforms. For example the MIDP persistence mechanism, the Record Management System, cannot directly serialise an object but can store an XML document directly as UTF-8 characters in a byte array.

An important decision for any type of mobile application is the decision about the trade-off between network traffic, memory usage, and storage space. It is an aspect of mobile computing that cannot be ignored as the application must carefully manage network traffic, as it costs the user for data transfer and potentially increases latency, memory is very limited, and persisting the data on the mobile device is not always feasible. Ideally the device would utilize its storage capacity as much as it could but because storage space limitations varies for different Java enabled phones the only realistic data that could be consistently persisted on the device would be user information (login and progress details) and some basic user course information, such as available courses, lessons, and the lesson content (essentially menu content).

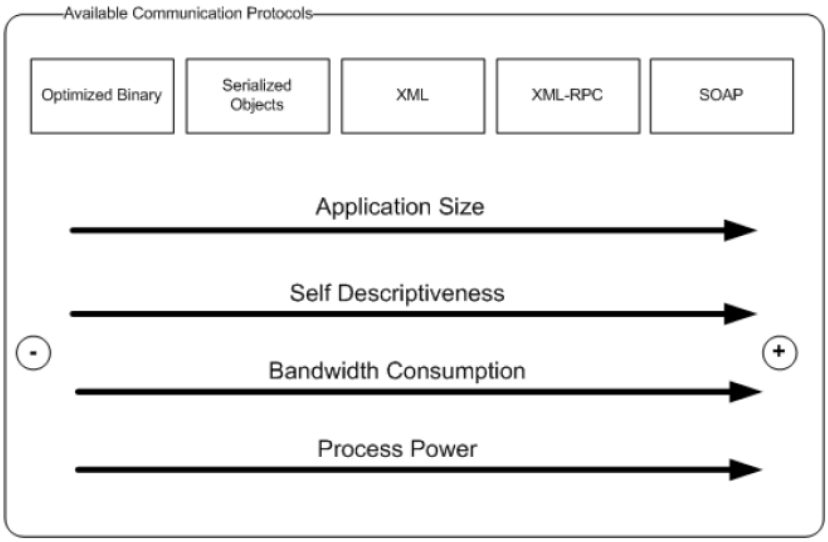

Figure 3: Communication protocol and trade-offs

THE USER INTERFACE

An emerging issue in the use of standard Java Micro Edition applications on the client is the limited sophistication of the user interface (UI) components included in the MIDP UI packages. As the presentational capability of mobile devices increases, working with the MIDP lowest common denominator approach will provide an interface that would be no more visually interesting than a simple thin client browser interface. Customised programming to take advantage of different devices' user interface capabilities would be problematic, due to the wide range of device capabilities on the market. To address these problems, a generic solution to enhancing a portable user interface for Java ME programs is required. One such solution is J2ME Polish (Virkus 2006), which includes an XML database of the capability of over 300 mobile devices and pre-processes the Java source code to inject the presentational polish. This tool supports a number of features, but in particular it supports a style sheet based approach to user interface presentation that enables Java ME programs to have a look and feel similar to style sheet base mobile browsers without sacrificing the power of the mobile client or the interoperability of the standard Java APIs. Figure 4 shows the same menu screen from the application running both with and without the J2ME Polish libraries, displayed on the emulator of the Sun Java Wireless Toolkit (Sun Microsystems 2006). The build specification for this application uses only generic J2ME Polish pre-processing, though device specific pre-processing is also possible, at the expense of platform neutrality.

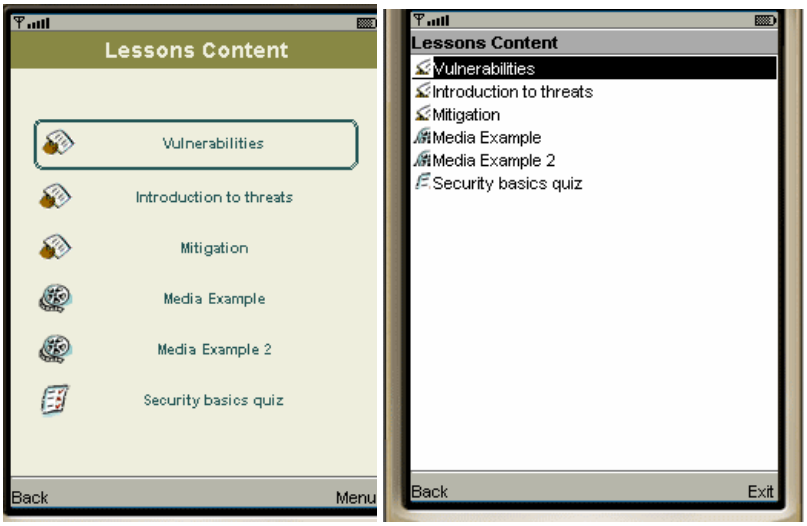

Figure 4: Generic J2ME user interface using J2ME Polish (left) and standard MIDP UI (right)

Similar device adaptive presentation can be achieved with thin client mobile browsers by using tools such as the WALL tag library (Passani 2006a). For content such as the menu example used here, such an approach would probably be more performant and easier to implement. However, the problem with markup is that it lacks the

ability of smart clients to execute applications, apart from some basic mobile browser plugins. In one respect, a combination of the browser and smart client approach would appear to offer the benefits of both architectures. Unfortunately this is problematic in practice because MIDlets cannot be seamlessly integrated into a browser workflow. In fact the only link between a MIDlet and a browser is that a web page can be used to download the MIDlet's JAD (application descriptor) and jar (archive) files. Once a MIDlet application has been downloaded, deployed and launched it operates independently of a mobile browser. This means that if we want to present the user with a seamless workflow and a consistent look and feel then even aspects such as menus need to be included in the smart client implementation.

TEST RESULTS

In this section we provide some test results based on access times through part of a use case, where the user navigates from the login page, though the course choice page, then selects the lesson and finally reaches different types of content. Testing was undertaken in four separate contexts. The first context was testing on a single node, where the server (an installation of JBoss application server 4) was running on the same machine as a Java ME client emulator. The second content was a remote server test, with a remote installation of JBoss being connected to over the Internet from a desktop machine running the Java ME client emulator. The third test was a 2.5G mobile phone (an iMate SP5 Windows Smartphone running the Tao Group 'intent' Java MIDlet Manager) connecting to the server over GPRS (General Packet Radio Service) via the Vodafone New Zealand mobile network. The final test was a 3G mobile phone (a Nokia N70) connecting to the server over 3GSM, again using the Vodafone New Zealand mobile network. In all of the tests the level of server logging was at 'info', which tended to slow the server's performance. However since the logging level was consistent for all tests this does not impact on the relative results. Test results for downloading video files over GPRS tended to vary considerably, such that a meaningful average time was difficult to ascertain, so we have record a minimum rather than an average.

Use Case Task	Local Server	Remote Server over network	Remote Sever over GPRS	Remote Server over 3G
Login	.5	.5	8	3
Text (menus)	3	4	5	4
Sound Media (6K)	3	4	6	6
Video Media (130K)	3	5	>15	12

Table 1: Average response times in seconds for different media types in different connection scenarios

The results of out tests were quite illuminating. There were two clear aspects to the figures. First, when we were transmitting small quantities of text, the differences between the different contexts were relatively small. This suggested that a significant part of the latency was in the XML processing, regardless of whether the client device was real or emulated and whether or not a mobile network was being used. The second clear issue was that as soon as we introduced multimedia content, the latency effect of the XML processing was overshadowed by the latency effect of downloading larger multimedia files, in particular video files, when connecting over the mobile phone network. This was not so significant for the short sound file that was used for testing, but was more marked for the video file. Since both of these clips were short (a few seconds each), this effect would be likely to be more marked for larger files, despite the use of buffering on the devices. The conclusions we drew from these tests were that (a) XML parsing in itself was an overhead but not excessive, (b) local caching of XML content would reduce latency effects considerably, (c) media files would have to be short, and recommended only for fast connections, and (d) these result emphasised that content should be available in more than one format, giving users appropriate options for their connection context.

FUTURE WORK

The current application serves a number of experimental content types but does not adapt itself to the differences between mobile devices. To cater for issues such as the excessive download time for movie clips on a 2.5G device, some adaptivity strategies need to be incorporated. This means providing some types of learning content in more than one format, so that (a) users with capable devices have a choice and (b) users with limited devices can access all content in at leats one format. Capability libraries such as WURFL can be useful here, since they are able to identify device capabilities from the user-agent HTTP header (Passani 2006b).

The User Profile section has not yet been fully implemented, but providing such feedback, and perhaps an adaptive interface based on user progress, would be valuable, As well as enabling the learner to reflect on their

progress, (including percentage of completion for available courses and current grading for each course), users could also to change their preferences, such as setting the default media types for certain content types. For example a user could choose to view still image slide shows rather than video to reduce download times and data volumes.

The current prototype is not being used in real learning situations, and has not been secured except by username and password. Before this prototype can progress beyond the proof of concept stage, a more secure communication layer will need to be provided, at least by using HTTPS transport, particularly if the system were to be used for formal assessment. Security is also an issue for the mobile device operating system. It is already clear that deployment on many devices is dependent on signing the Java code with verified security certificates.

A related system that uses adaptive mark-up rather than a smart client is described in (Parsons and Schroder 2006). A more detailed test that compared the performance of these two approaches would be a useful future study.

CONCLUSIONS

In this paper we have described the architecture of a mobile learning platform based on XML messaging between a server and a smart Java client. We have described the general architecture, the reasons for choosing this architecture and some performance statistics that indicate the potential latency effects of using an XML messaging/parsing architecture. The prototype system has demonstrated that a smart client Java ME application, connected to a server using XML, can provide a rich, interactive environment for the mobile learner, However there are certain usability issues associated with certain types of mobile device. Our experiments showed that certain types of content could not be delivered using a GPRS device within a reasonable time. Some of these problems may be addressed by various forms of optimisation to reduce network traffic, reduce latency, and utilize the mobile device's own storage and processing capabilities as much as possible. Network traffic may be reduced by using techniques such as data compression and caching, while latency may be reduced by including some background processing for downloading. In both cases the mobile device can be leveraged to store local data (caching) and take on more processing responsibility. For example, large documents might be downloaded and cached that would support a sequence of user interactions without further reference to the server. Work is continuing on the prototype to address these technical issues before testing can commence with users.

REFERENCES

Casalino, N., D'Atri, A., Garro, A., Rullo, P., Saccà, D. and Ursino, D. 2006, Proceedings of IEEE International Conference on Mobile Communications and Learning Technologies: An XML-based multi-agent system to support an adaptive cultural heritage learning,, pp. 224.

Chang, V. 2006, Proceedings of IEEE International Conference on Mobile Communications and Learning Technologies: Web Service Testing and Usability for Mobile Learning, IEEE, pp. 221.

Coulton, P., Rashid, O., Edwards, R. and Thompson, R. 2005, 'Creating Entertainment Applications for Cellular

Phones' ACM Computers in Entertainment, vol. 3, no. 3.

Domer, J., Nanja, M., Srinivas, S. and Keshavachar, B. 2004, Proceedings of 2004 workshop on Interpreters, Virtual Machines and Emulators (IVME'04): Comparative Performance Analysis of Mobile Runtimes on Intel XScale® Technology, ACM Press, Washington, D.C., USA.

Fowler, M. 2003, Patterns of Enterprise Application Architecture, Addison-Wesley, Boston.

Graham, I. 2003, A Pattern Language for Web Usability, Addison-Wesley, London.

Hibernate 2006, Relational Persistence for Java and .NET, Hibernate, Available: http://www.hibernate.org/, Accessed: June 16th 2006

Hjelm, J. 2000, Designing Wireless Information Services, Wiley, New York.

Koschembahr, C. V. and Sagrott, S. 2005, In Mobile Learning: a handbook for educators and trainers, (Eds, Kukulska-Hulme, A. and Traxler, J.) Routledge, London.

Mahmoud, Q. H. 2005, J2ME Luminary: Mark Duesener of Vodafone Group, Sun Microsystems, Available: http://developers.sun.com/techtopics/mobility/midp/luminaries/markduesener/, Accessed: July 3rd 2006

Meisenberger, M. 2004, Mobile Learning Engine, Available: http://drei.fh-joanneum.at/mle/start.php?sprache=en&id=0, Accessed: July 3rd 2006

MySQL AB 2006, MySQL 5.0 Relational Database, Available: http://www.mysql.com/, Accessed: June 16th 2006

Ortiz, C. E. 2003, The Generic Connection Framework, Available: http://developers.sun.com/techtopics/mobility/midp/articles/genericframework/, Accessed: July 3rd 2006

Parsons, D. and Schroder, S. 2006, Proceedings of 11th United Kingdom Academy of Information Systems Conference: Adaptive Information Systems for the Web 2.0: Developing A Mobile Learning Architecture, UKAIS, Cheltenham, U.K.

Passani, L. 2006a, WALL - The Wireless Abstraction Library, Available: http://wurfl.sourceforge.net/java/wall.php, Accessed: July 3rd 2006

Passani, L. 2006b, The Wireless Universal Resource File, Available: http://wurfl.sourceforge.net/, Accessed: July 3rd 2006

Sakkopoulos, E., Lytras, M. and Tsakalidis, A. 2006, 'Adaptive Mobile Web Services Facilitate Communication and Learning Internet Technologies' IEEE Transactions On Education, vol. 49, no. 2.

Slomiski, A. 2004, On Using XML Pull Parsing Java APIs, Available: http://xmlpull.org/history/index.html, Accessed: June 16th 2006

Sun Microsystems 2006, J2ME Wireless Toolkit, Sun Microsystems, Last accessed

Virkus, R. 2006, J2ME Polish, Available: http://www.j2mepolish.org/index.html, Accessed: June 16th 2006

WAP Forum 1999, Wireless Application Protocol: Wireless Markup Language Specification Version 1.1, Wireless Application Protocol Forum Ltd., http://www.wapforum.org/what/technical/SPEC-WML-19990616.pdf, Last accessed

WindowsForDevices.com 2006, Microsoft to lead in advanced mobile phones by 2010, Available: http://www.windowsfordevices.com/news/NS7072037463.html, Accessed: June 15th 2006

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of Massey University's Albany Strategic Research Fund, the Institute of Information and Mathematical Sciences and Vodafone New Zealand in funding this work.

COPYRIGHT